



Rule Based Modeling and Model Refinement

Elaine Murphy, Vincent Danos, Jérôme Feret, Jean Krivine, Russell Harmer

► To cite this version:

Elaine Murphy, Vincent Danos, Jérôme Feret, Jean Krivine, Russell Harmer. Rule Based Modeling and Model Refinement. H. Lodhi and S. Muggleton. Elements of Computational Systems Biology, John Wiley & Sons, pp.83–114, 2010, Wiley Book Series on Bioinformatics, 978-0470180938. <10.1002/9780470556757.ch4>. <inria-00527981>

HAL Id: inria-00527981

<https://inria.hal.science/inria-00527981v1>

Submitted on 16 Feb 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



HAL Authorization

Rule Based Modelling and Model Refinement

Elaine Murphy*	Vincent Danos	Jérôme Féret
University of Edinburgh	University of Edinburgh	ENS-INRIA-CNRS
Jean Krivine	Russell Harmer	
Harvard University	CNRS-Paris-Diderot	

August 12, 2010

Rule-based modelling is an effective way of handling the explosive combinatorics of biological networks. The use of partial objects in describing molecular interactions means that only the necessary conditions for a rule are specified and not the complete chemical entities taking part in a reaction. This leads to descriptions that are easier to set up and more compact. Networks of substantial scale can be described without having to reduce the combinatorics of the system—as other approaches must.

An important aspect of the rule-based approach is its agility, as one can easily modify rules to incorporate new knowledge, or test different assumptions. A special and rather frequent case is when one wishes to replace a rule with ones imposing stronger conditions. This process is called *refinement* and we approach it in this study both from the practical and the theoretical point of view.

There are various reasons why one would like to use refinement:

- one wants to understand how the activity of a rule varies with its application contexts
- or one realizes that more conditions are necessary than previously thought
- or more subtly one wishes to evolve the behaviour of the current system

The notion of behaviour-preserving, or neutral, refinement commands an analysis of the possible symmetries of partial complexes. Here we need a rigorous algebraic theory to see through the intricacies caused by symmetries. Incidentally, the problem of neutral refinement is one of a family of problems that is well-studied in the theory of concurrent systems, usually under the catch phrase of ‘behavioural equivalence’. The form of equivalence we are looking for here is especially strong since it should hold irrespective of the other rules defining the dynamics of the model.

The material is organized as follows. We begin with a brief introduction to the Kappa language (§1). Next, we present several examples (§2) of refinements. We have, in particular, a somewhat lengthy example that shows how refinements can be used to evolve complex behaviour from simple systems. By introducing mutant variants of agents that alter the behaviour of a single rule, it is possible to change dramatically and in unexpected ways the outcome of a pathway (§2.2).

Once we are reassured that the notion of refinement is actually useful, we turn to the second part, namely the mathematical development of rule refinement. An algebraic version of (a mild simplification of) Kappa is introduced (§3). This is framed in basic category theory which allows us to make use of existing mathematical techniques. Previous work in this area developed a framework

*The first author was the principal contributor to this study, others have contributed equally.

for homogeneous rule refinement where agents of the same type had the same sets of sites [1]. The framework developed here is much more general and introduces the notion of addresses to access specific agents in partial complexes (§4). This enables us to model a much larger class of rule refinements and an example is given of a model that could not have been dealt with previously. We end by deriving a general formula for neutral refinement and show that the stochastic transition system underlying the rule set is unchanged.

The following is self-contained. Nevertheless, readers might want to consult earlier Kappa references on: a concrete example of the agility of rule-based modelling [2], the use of debugging methods based on abstract interpretation [3], the development of techniques for large scale stochastic simulation [4], or the study of statistical asymptotic properties of simple Kappa networks [5].

1 Kappa, briefly

The realm of protein-protein interactions commands a picture which is substantially different from the traditional closed biochemical world of metabolic networks. The innumerable combinations that non-covalent binding brings about all but forbid an extensional view of protein networks. An analogy with group theory might help. Even when a group is finite, its multiplication table might be so large that no extensional description is possible. Yet, in some favourable cases, one can handle such groups using generators and relations. Accordingly, in Kappa and other rule-based languages such as BNG [6], agents do not represent species (aka complexes) but their elementary components, that is to say proteins. Agents are used as generators and rules specify how more complex objects can be assembled from them. This does not preclude simulation, and one can even simulate systems with an infinite number of possible complexes (supposing agents are available in infinite numbers of course). For unconditional rule sets, by using simple statistical mechanics techniques, one can derive conditions on the affinities and copy numbers of agent types for the set of possible unique complexes to become infinite in the limit of infinite populations; so to some extent one can predict where traditional enumerative methods cease to apply [5].

Throughout this study we will use the rather generic term “agent” to designate our basic entities, but in most applications these are indeed idealizations of proteins. Agents have sites which can be used to bind to other sites, and can also hold an internal state. Binding via sites accounts for the formation of domain-mediated complexes, while internal states account for post-translational modifications such as phosphorylation. (Syntactically, as we will soon see in the examples below, the binding of two sites is represented by a common superscript, while the internal state of a site is indicated as a subscript.) Different types of atomic events can be combined in a rule such as: binding, unbinding, modification of an internal state, addition or deletion of an agent. A binding rule requires two distinct free sites and hence, at any given time, it is not possible for a site to be bound more than once, although an agent may be bound simultaneously on different sites.

A Kappa model consists of: 1) a rule set specifying how the initial solution may evolve, with each rule being given a rate, and 2) an initial state that which declares the names, sites and copy numbers of all agents present in the system at the outset. Each rule in the rule set has a likelihood attached to it which gives the model a stochastic behaviour. This likelihood is proportional to the number of ways in which the rule can be applied to the current state of the system multiplied by its rate. This rate is a measure of how efficient a rule is at turning a chance encounter of reagents into an actual reaction. In the case where agents have no sites at all, a model boils down to a Petri net and the dynamics is the same as the mass action law put in Gillespie form [7].

2 Refinement, practically

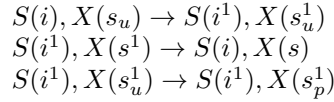
2.1 A Simple Cascade

In order to introduce our notation for rules and agents, and demonstrate the notion of refinement in a first simple case, we start with an elementary cascade. This type of biological circuitry occurs frequently in actual pathways (e.g. see Ref. [8]).

In our example we have one kinase S , covalently modifying another kinase X , which in turn modifies some third agent Y . Each agent type is supposed to have a single site, and the sites of X and Y hold an internal state of either u (unphosphorylated) or p (phosphorylated); one says X , Y are active when they are phosphorylated. To keep things simple the model does not include any mechanism to deactivate X or Y .

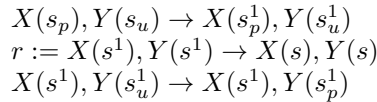
2.1.1 The rules

The interactions between S and X are defined by the following rules:



In this rule set, a binding is represented by a shared exponent, eg $S(i^1), X(s^1)$ represents a binding between the S and the X agents via their respective i and s sites. The first rule in the triplet specifies the conditions needed for such a binding to take place: one needs the sites i and s to be free, and one also needs the site s to have a specific internal state u , indicated as a subscript s_u . One might say that S is ‘smart’ in so far as it does not bind a target that is already modified, ie of the form $X(s_p)$. The second rule represents the unbinding of the two molecules. Contrary to the first one, this rule does not depend on the s site of X being in a particular internal state. The ability to not have to specify the entirety of the context in which an event can be triggered—which we alluded to earlier, and which is sometimes called the ‘don’t care, don’t write’ convention—already shows here in a very simple form. The third rule represents the activation of X , ie the change of X ’s internal state from u to p .

A second and similar rule triplet defines the interactions of X and Y :



This rule set only differs from the previous one only in that the X agent is required to have a phosphorylated s site in order to bind a Y agent, as stipulated in the first rule. This ensures that the first half of the cascade happens before the second and, in particular, that Y cannot be activated if there is no S signal.

2.1.2 On the importance of being off

To complete the definition of our model and define a proper stochastic system, we need to choose rates for the above rules, and to define an initial state. We will assume that all rates are 1, except the rate k of the rule r whereby X and Y detach. In fact, we want to ask a classical cascade question,

namely how the overall rate of production of the active (phosphorylated) form of Y depends on k . (One could ask the same for the rate at which S and X detach.) Intuitively, if k is too low, X will tend to remain attached to Y for long periods of time after Y has been activated and during which it will be unable to interact with any other Y ; moreover, this also potentially prevents the Y to which it is attached from further propagating the signal. On the other hand, if k is too high, X will detach from Y before it has had a chance to activate it. The performance of the cascade is therefore a non-monotonic function of k , and somehow an efficient activation of Y needs to strike the right balance between binding too loosely and too tightly.

Let us demonstrate this numerically by varying k . Suppose one starts with an initial state of $15 * S(i) + 60 * X(s_u) + 120 * Y(s_u)$, so as to have significantly more Y s than X s (otherwise the stickiness effect will be largely invisible). As expected, the activation of Y is rather slow when $k = 0.1$ (Fig. 1), becomes faster for $k = 10$ (Fig. 2), and slows again when $k = 100$ (Fig. 3).[‡]

Specifically the Y 's rise times, defined as the time where half of the Y s are active, are respectively and approximately 8, 5, 7. A closer numerical examination of the rise time of Y as a function of k would reveal the boundaries of the control area which produces the optimal behaviour of the rule. With the above set of parameters, and for k within $[1, 50]$ one has a rise time below 6. Outside of this interval the production slows down either because binding is too sticky or too liquid. We make a mental note that there is a rather large interval k where the cascade operates fast; how large of course will depend on all the other parameters as well.

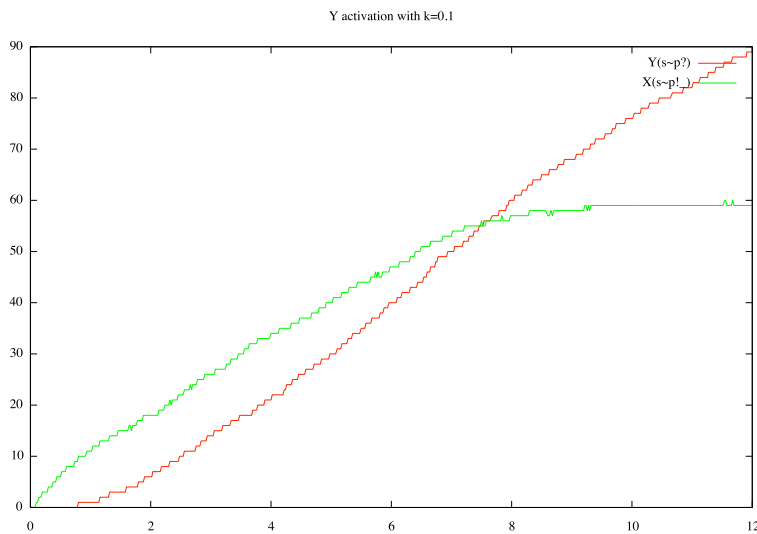


Figure 1: Simulation with $k = 0.1$ (sticky case): Y 's rise time is approximately 8 (higher curve), at which time about 100% X s are active and bound (lower curve).

[‡]Simulations are obtained with an implementation of the Kappa language which can be obtained at kappalanguage.org.

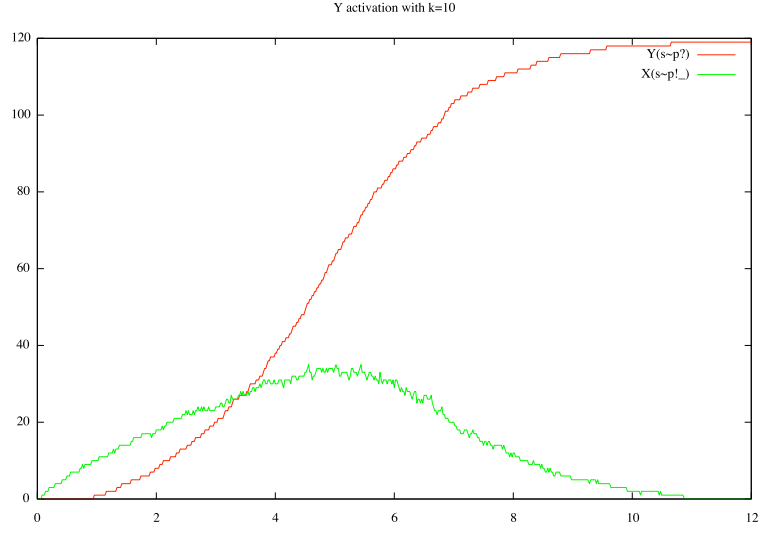


Figure 2: *Simulation with $k = 10$ (near optimum): Y 's rise time is about 5 (higher curve), at which time about 50% of X s are active and bound (lower curve).*

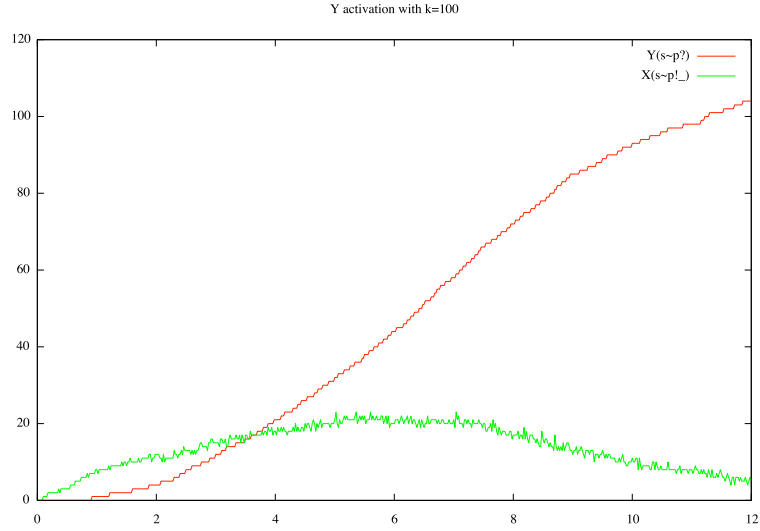


Figure 3: *Simulation with $k = 100$ (liquid case): Y 's rise time is about 7 (higher curve), at which time less than 30% of X s are active and bound (lower curve).*

2.1.3 Refinement of the off rule

However, there is another method to optimize the behaviour of this rule without having to investigate the effect of varying the rate parameter to probe the control area. In this system, the rule r

regarding the detachment of X from Y given above is applied regardless of the internal state of Y . A natural strategy to optimize the cascade is to make r depend on Y 's internal state, and split it into two subcases with respective rates k_1 and k_2 :

$$\begin{aligned} r_1 &:= X(s^1), Y(s_u^1) \rightarrow X(s), Y(s_u) \\ r_2 &:= X(s^1), Y(s_p^1) \rightarrow X(s), Y(s_p) \end{aligned}$$

The substitution of r with r_1, r_2 is an example of a rule refinement. If one sets $k_1 := k_2 := k$, the behaviour of the system will remain unchanged. Such a case where a refinement does not alter the dynamics of a system is what we have called earlier a neutral refinement. On the other hand, if one sets $k_1 := 0$, so that X never detaches if the Y site has not been activated, and $k_2 := \infty$, ie X detaches as soon as activation has occurred, one will clearly accelerate the activation of Y . In effect, in this example splitting r is exactly what one needs to make Y 's rise-time a monotonic function of both k_1 (decreasing), and k_2 (increasing), whereas before, as we have seen above, it was not a monotonic function of k . It follows that the optimal assignment for k_1 and k_2 is as above. This definitely changes the dynamics, and constitutes what we have called a kinetic refinement.

2.2 Another cascade

Here is another two-tiered cascade:

$$\begin{aligned} &S(i), X(s_u) \rightarrow S(i^1), X(s_u^1) \\ r &:= S(i^1), X(s^1) \rightarrow S(i), X(s) \\ &S(i^1), X(s_u^1) \rightarrow S(i^1), X(s_p^1) \\ \\ r' &:= X(s_p^?, y), Y(s_u) \rightarrow X(s_p^?, y^1), Y(s_u^1) \\ &X(y^1), Y(s^1) \rightarrow X(y), Y(s) \\ &X(y^1), Y(s_u^1) \rightarrow X(y^1), Y(s_p^1) \end{aligned}$$

The first rule triplet is identical to that in the first example. However, the second rule triplet has an important difference. First, X now has two sites s and y , one to bind S , and the other to bind Y .² Second, the X/Y association rule r' does not require X to detach from S to attach to a Y . The superscript ? on the s in X precisely expresses that the X, Y association rule does not care whether the X agent is free or bound on its other site s . The rule does however ask X to be activated as in the first example.

These rules are also more solid examples of the ‘don’t care, don’t write’ convention of which we have seen simple examples earlier (§2.1.1). The left hand sides of rules are partial complexes containing only the conditions that are necessary for the rule to be activated. For example, in the fifth rule, the s site of the X molecule is not mentioned because we do not care what internal or binding state it is in. Every such rule embodies a regularity assumption which may or may not hold. One use of refinements is to allow one to revise such assumptions and admit in the dynamics less regularity, viz more dependence on the context of a rule, than hitherto assumed. However, with this second example we wish to explore another aspect of refinement, namely that it enables complex behaviour to evolve from simple systems. We are going to perform a series of refinements of the above cascade and obtain rather surprising behaviours.

²This division of a site can be seen as another kind of transformation.

2.2.1 Neutral refinements

Our first refinement concerns r the S/X dissociation rule, which we replace with the following two:

$$\begin{aligned} r_1 &:= S(i^1, s_1), X(s^1) \rightarrow S(i, s_1), X(s) \\ r_2 &:= S(i^1, s_2), X(s^1) \rightarrow S(i, s_2), X(s) \end{aligned}$$

This refinement has the effect of adding a new site s to S with an internal state that can be either of 1, 2.³ As there are no rules that modify this internal state, the refinement can be seen as defining a variant of S . To use a more biological terminology, we will say that $S(i, s_1)$ and $S(i, s_2)$ are isoforms, sometimes simply written S_1, S_2 hereafter.

Our second refinement concerns r' the X/Y association rule, which we replace with the following two:

$$\begin{aligned} r'_1 &:= X(s_p^?, y), Y(s_u, y_1) \rightarrow X(s_p^?, y^1), Y(s_u^1, y_1) \\ r'_2 &:= X(s_p^?, y), Y(s_u, y_2) \rightarrow X(s_p^?, y^1), Y(s_u^1, y_2) \end{aligned}$$

This time, we have introduced isoforms of the Y agent, using the new site y , which we again write simply Y_1, Y_2 .

For our final refinements we pick r'_1 and r'_2 and make a distinction based on whether the s site of the X agent is bound or free:

$$\begin{aligned} r'_{11} &:= X(s_p^-, y), Y(s_u, y_1) \rightarrow X(s_p^-, y^1), Y(s_u^1, y_1) \\ r'_{12} &:= X(s_p, y), Y(s_u, y_1) \rightarrow X(s_p, y^1), Y(s_u^1, y_1) \\ r'_{21} &:= X(s_p^-, y), Y(s_u, y_2) \rightarrow X(s_p^-, y^1), Y(s_u^1, y_2) \\ r'_{22} &:= X(s_p, y), Y(s_u, y_2) \rightarrow X(s_p, y^1), Y(s_u^1, y_2) \end{aligned}$$

Here the s^- superscript carried by s means that it is bound but the rule does not care what it is bound to. In fact, in this simple example s at X can only be bound to one of the two S isoforms, so this is just a convenient abbreviation, and one could equivalently write for example:

$$r'_{21} := S(i^1), X(s_p^1, y), Y(s_u, y_2) \rightarrow S(i^1), X(s_p^1, y^2), Y(s_u^2, y_2)$$

This begs the remark that a left hand side can have any number of agents, in any configuration. The important thing to notice is that neither r'_{11} , nor r'_{21} know which isoform of S they are dealing with.

Applying these four successive refinements gives us a new rule set, where r is replaced with r_1, r_2 , and r' with $r'_{11}, r'_{12}, r'_{21}$, and r'_{22} . We now have two inputs S_1, S_2 , and two outputs Y_1, Y_2 that we can manipulate independently. The question we ask now is another classical cascade question, namely we wish to understand the range of input/output dependencies one can attain with our cascade.

2.2.2 Kinetic refinements

If all refined rules were to inherit the original rule rates, this would be a neutral refinement, all our isoforms would be indistinguishable, and there would be no interesting dependency created by our rule set. We need to alter some of the rule rates to obtain a kinetic refinement with an interesting behaviour. This is what we do now.

³To make this addition completely uncontextual one could write explicitly $s_1^?, s_2^?$ in the two rules above.

We first set r_2 to have a lower rate and r_1 to have a higher rate than the original rule r . All other rates being equal, this has the effect of making S_2 stickier, and S_1 more liquid. As S_1 's dissociation rate is high, it is less likely to stay bound to X at s . So with only S_1 s in the system one would expect to see higher numbers of *s-free* active X , compared with the neutral refinement. Conversely, the S_2 variant is more likely to stay bound to X at s , and one would expect to see higher numbers of *s-bound* active X , with only S_2 s in the system. Furthermore, as we have seen in the first example (§2.1.2), for well-chosen values of the various cascade parameters, one has a large window of efficient dissociation rates (our numerical study concerned the dissociation rate of the second tier of the cascade, but the same holds for the first one). So our variant signals S_1 and S_2 may well be equally efficient, and in fact nearly optimal in their activation of Y s. They only differ in style, or in their degree of liquidity.

Now, we can take advantage of this difference in the transient behaviour of our signal isoforms by setting the rates of r'_{11} and r'_{22} to 0. This amounts to requiring that an *s-bound* active X can attach only to the Y_2 variant, whereas *s-free* active X can attach only to the Y_1 variant.

With this second kinetic refinement in place, and following the informal argument above, one would now expect S_1 to activate primarily Y_1 , and S_2 , Y_2 . To confirm this, we can run simulations varying the numbers of S_1 , S_2 :

$$\begin{aligned} &15 * S(i, s_1) + 0 * S(i, s_2) + 50 * X(s_u) + 120 * Y(s_u, y_1) + 120 * Y(s_u, y_2) \\ &0 * S(i, s_1) + 15 * S(i, s_2) + 50 * X(s_u) + 120 * Y(s_u, y_1) + 120 * Y(s_u, y_2) \end{aligned}$$

In the first case, one has only the S_1 isoform, and this has the effect of mainly producing active Y_1 s (Fig. 4), while in the second one (Fig. 5), one has only the S_2 isoform, and this has the effect of mainly producing active Y_2 s. As we have no rule for the deactivation of X s, all X s eventually become free at s (because the S/X association rule tests for the internal state of X), which explains why some active Y_1 is also produced in the second case. However, this happens at a much slower rate than the production of active Y_2 . Were we to include deactivation rules for X or Y , we would see a low production of active Y_1 at all times. The design we obtain successfully establishes a specific input/output relation whereby S_1 only activates Y_1 , and S_2 only Y_2 , despite the fact that both cascades share a component, namely X .

Of course, if one reflects on the design, there is no magic, as it hinges on the shared component reflecting its binding status upstream in its behaviour downstream (the role of the second kinetic refinement). The subtle point is that it does so without sensing the upstream isoform it is bound to—the information is contained only in the different residency times of the two signal isoforms (the role of the first kinetic refinement). This *in numero* experiment shows that there is ‘plenty of room’ at the relevant time scales (to paraphrase a famous quote from Feynman).

2.2.3 Signal integration

Thus, and so far, the results of our numerical experiment are as one would expect given the rule set. It is when we ask what happens if both signal isoforms of S are present in the system at the same time that one gets a surprise. If there is an excess of available X , both active Y_1 and Y_2 are produced at approximately even rates, and the system behaves as two independent juxtaposed pathways (Fig. 6). However, if one has a limited number of X s as in:

$$15 * S(i, s_1) + 15 * S(i, s_2) + 10 * X(s_u) + 120 * Y(s_u, y_1) + 120 * Y(s_u, y_2)$$

the S_2 to Y_2 pathway takes over, and despite the presence of S_1 , only active Y_2 is significantly produced (Fig. 7). The behaviour of the system in this case can be seen to be analogous to that of

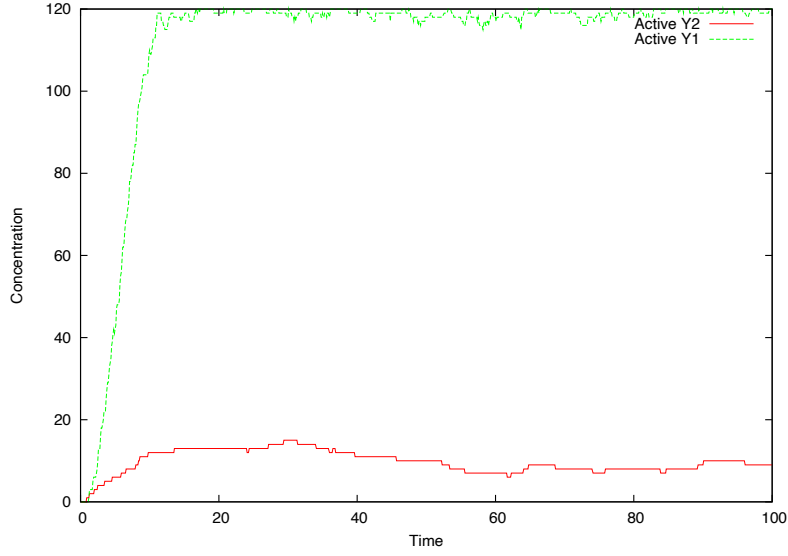


Figure 4: *Simulation with only S_1 : Y_1 gets activated, and not Y_2 . Here and in the following, rules for spontaneous inactivation of X, Y at a rate of 0.01 were added.*

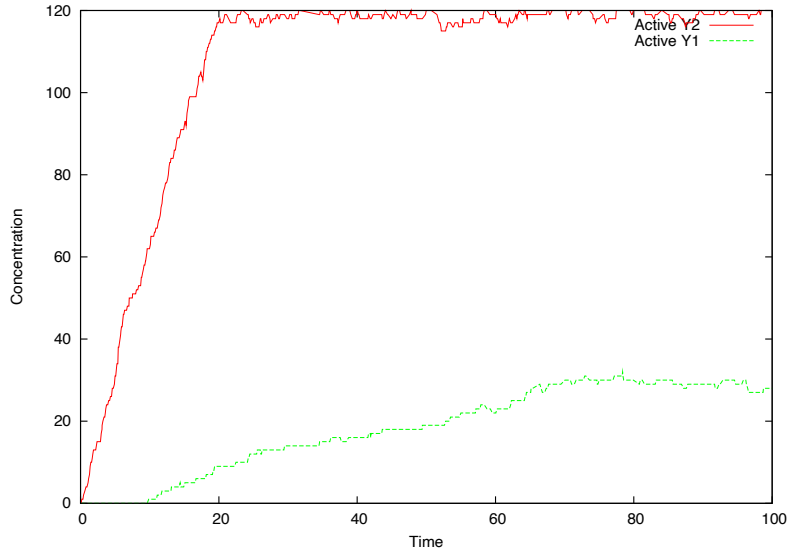


Figure 5: *Simulation with only S_2 : Y_2 gets activated, and not Y_1 .*

a transistor whereby S_2 can completely override S_1 . To summarize the situation, and despite the fact that it is always tricky to give Boolean representations of biological circuitry, one could say that the final design behaves as the Boolean function $Y_1, Y_2 = S_1 \wedge \neg S_2, S_2$. Note however that the amount of control exerted by S_2 can be modulated by the amount of the shared component X , as

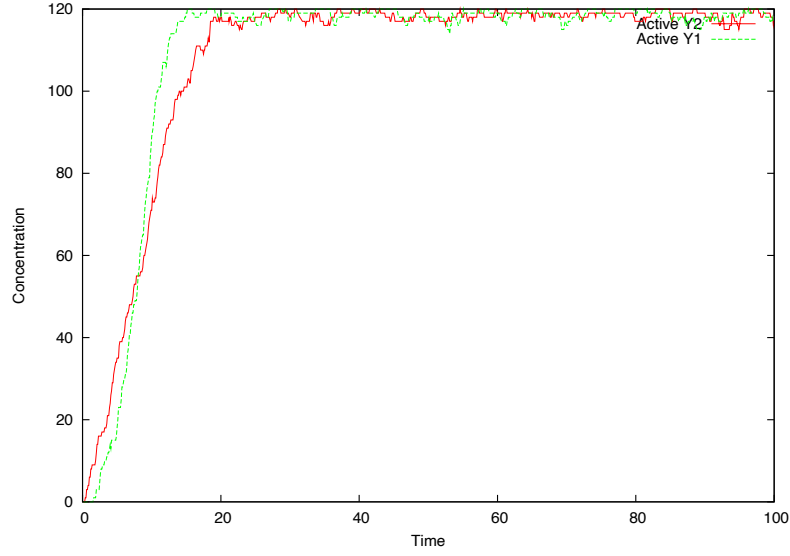


Figure 6: *Simulation with both S_1 and S_2 and an excess of X : both Y_1, Y_2 get activated at the same rate.*

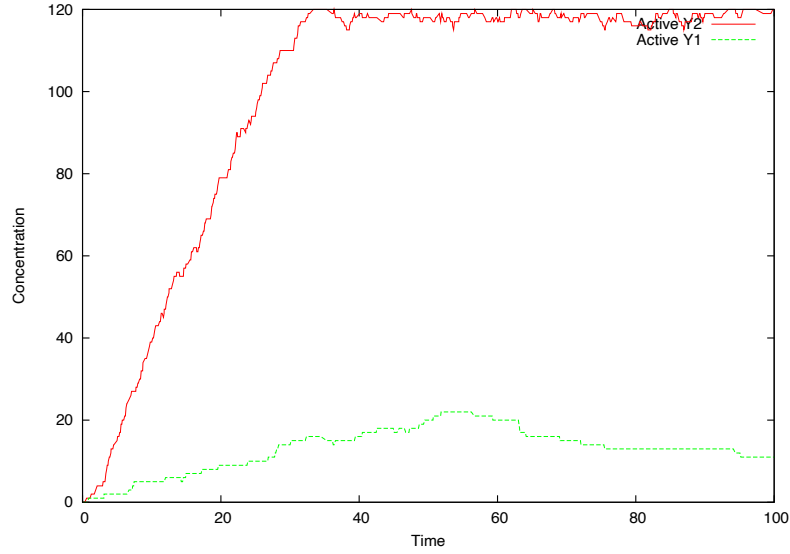


Figure 7: *Simulation with both S_1 and S_2 and few X s: S_2 overrides S_1 , as only Y_1 gets significantly activated.*

when X is in excess the mapping becomes simply the identity (Fig. 7).

2.2.4 Discussion

One can easily imagine that such a ‘transistor’ circuit as we have just derived could be useful for cellular decision-making. Of course, this prompts the fascinating question of whether it can be implemented, and more specifically, whether a variation in a real protein cascade, mimicking our derivation of the said circuit, can be engineered by varying the code of the various relevant proteins. This is one of a set of questions that is being actively researched in the new and exciting engineering field of synthetic biology (see [9] for a recent review). One might soon be able to answer this kind of question, but at the moment the technology of synthetic protein networks is far less advanced than that of transcriptional circuits.

Our design also raises the intriguing possibility of encoding information in the transient assembly of complexes, a mode of information processing that is unique to the world of protein interactions, and which needs a language where binding is a primitive operation to be comfortably dealt with. From the specific point of view of this study, it is also an example of the explanatory power of refinements, as they make the design particularly transparent, whereas presumably it would be rather difficult to understand it without resorting to refinements.

This second extended example concludes the motivational and methodological part of our refinement study. By now hopefully the reader will have a good intuition of refinements and their use. So we turn now to other, less biologically-motivated examples, no longer meant to show the usefulness of refinements but rather to anticipate some of the mathematical challenges we will need to address later and prepare the ground for the algebraic treatment that constitutes the core of this study, and which comes next (§3 onwards).

2.3 The SSA convention

The previous examples show how natural and intuitive refinement can be but have maybe given the (mistaken) impression that obtaining a neutral refinement is a fairly straightforward matter. In fact, this is far from being the case in general and there are several subtleties that we need to take into account.

Let us begin examining this by considering a straightforward binding rule $r := A(x), A(x) \rightarrow A(x^1), A(x^1)$ between agent type A (with at least one site x) and itself, with stochastic rate constant k_r . Now, in the case where rule r is actually a reaction, ie A has just the one site x with no internal state (cf. the ‘don’t care, don’t write principle’), it is usual, during stochastic simulation, to calculate the activity of the rule as $k_r n(n-1)/2$, where n is the number of A s currently unbound on site x [7]. In this convention, call it the SSA convention, an event is the identification of a multiset, here an unordered pair of A s, and there are ${}^nC_2$ such unordered pairs not n^2 .

The SSA convention works well for reactions (aka Petri nets, or multiset rewriting), because any application of a reaction preserves the reaction symmetries (as expressed in its multiset representation). This is no longer true with our richer notion of rewriting where the local symmetries of a rule may increase or decrease depending on its application context. So we shall use another convention where an event is an injection of the rule left hand side, and one does not attempt at quotienting those injections by the rule automorphisms (aka symmetries).

To understand this better, let us examine a concrete case where r is not a reaction. This means that the rule can instantiate as several distinct reactions. For example, let us suppose that the site x actually has an internal state, blue or red, that the rule does not mention. Intuitively, the result of firing r is the creation of either a red-red, a red-blue or a blue-blue pair of A s. We can make this

explicit with the following refinement of r into the three subcases:

$$\begin{aligned} r_1 &:= A(x_R), A(x_R) \rightarrow A(x_R^1), A(x_R^1) \\ r_2 &:= A(x_R), A(x_B) \rightarrow A(x_R^1), A(x_B^1) \\ r_3 &:= A(x_B), A(x_B) \rightarrow A(x_B^1), A(x_B^1) \end{aligned}$$

Let us assume that these three refined rules are actually reactions, ie we have now revealed everything about A 's sites and states. Notice that r_2 , unlike the other two refinements, breaks the symmetry of the original rule r . This has a significant consequence on how we calculate its activity; unlike r_1 and r_3 that follow the above pattern, ie $k_{r_1} n_R(n_R - 1)/2$ and $k_{r_3} n_B(n_B - 1)/2$, r_2 's activity is $k_{r_2} n_R n_B$. The total combined SSA activity of r_1 , r_2 and r_3 is the same as that of r :

$$k_r n(n - 1)/2 = k_r n_R(n_R - 1)/2 + k_r n_B(n_B - 1)/2 + k_r n_R n_B$$

where $n = n_R + n_B$. In other words, setting $k_{r_1} := k_{r_2} := k_{r_3} := k_r$ defines a neutral refinement.

Now, what about the other convention, where one does *not* take into account a rule's apparent local symmetry when calculating its activity, ie does not divide by the number of automorphisms (in r 's case, 2). Obviously, we have to divide r 's rate constant by the number of automorphisms in order to recover the original behaviour, in our case assigning $k_r/2$ to r . This keeps the activity of r unchanged $k_r/2 n(n - 1)$, and we readily calculate that

$$(k_r/2) n(n - 1) = (k_r/2) n_R(n_R - 1) + (k_r/2) n_B(n_B - 1) + (2k_r/2) n_R n_B$$

so that this time we need the assignment $k_{r_1} := k_{r_3} := k_r/2$ and $k_{r_2} := k_r$ to get a neutral refinement. It seems here we need a non-uniform assignment of rate constants in order to obtain a neutral refinement, and the reader might think the SSA convention is more natural. It is not. The non uniform rate assignment is an optical illusion since one can rewrite the above as:

$$(k_r/2) n(n - 1) = (k_r/2) n_R(n_R - 1) + (k_r/2) n_B(n_B - 1) + (k_r/2) n_R n_B + (k_r/2) n_R n_B$$

where all rates are now equal, and one has two copies of the asymmetric refined rule r_2 . These copies are associated to the two non isomorphic injections of the original r left hand side in r_2 's. The subtle point is that although the injections are not isomorphic (since they map the first A either to a red or a blue type; see also the discussion in §3.4, §4.1), the obtained rules are. Likewise one can inject r 's lhs in two ways in r_1 's, and r_3 's but this time the injections are isomorphic (meaning they are conjugated by an automorphism of their joint target) since both extensions preserve r 's automorphisms. So in these two latter cases, there is no need for a copy. This point of view leads indeed to our main technical result (Th. 1, §4.3) where all rates are kept equal to the original rule -provided one uses our convention, not the SSA one.

In summary, in a rule-based setting we can either rely on the SSA convention (underlying Gillespie's stochastic simulation algorithm) or the one presented above. The calculation of the rate constants required to obtain a neutral refinement will be affected. Our choice is more natural in that it brings a general result of a simpler form. In this red-blue example, both conventions seem to be on a par, the next example will show a case where the SSA convention is less natural.

2.4 A less obvious refinement

Let us consider a final example which should demonstrate the full subtlety of determining the appropriate rate constants for a neutral refinement. To this end, consider two agent types B and

C , again each with only one site x , and define a family of systems $x(n_1, n_2)$ consisting of n_1 free $C(x)$ and n_2 dimers $C(x^1), B(x^1)$.

Consider the rule $r := C(), B() \rightarrow C()$ with rate 1.0. Note that r does not mention x at all. This means that r can be applied irrespective of the binding of B and C . Whatever the bindings are, the effect of the rule will be, provided $n_2 > 0$, to delete a B and to bring $x(n_1, n_2)$ to a new state $x(n_1 + 1, n_2 - 1)$. If $n_2 = 0$ then there are no B s left in the system and the rule cannot be applied, in which case we say that the system is in deadlock.

We would like to refine r into mutually exclusive subcases depending on the bindings between B and C . Specifically we want to use the following three refined rules.

$$\begin{aligned} r_1 &:= C(x^1), B(x^1) \rightarrow C(x) \\ r_2 &:= C(x^1), B(x^1), C(x^2), B(x^2) \rightarrow C(x^1), B(x^1), C(x) \\ r_3 &:= C(x), C(x^1), B(x^1) \rightarrow C(x), C(x) \end{aligned}$$

Note that, by deleting the agent B , we implicitly also delete all bonds emanating from that B so, in particular, the previously bound C becomes free. Each of these is a particular case of r in the sense that their left hand sides embed, sometimes in more than one way, that of r . Intuitively, r_1 is the case where B and C are bound together, r_2 is the case where B and C are both bound but not to each other, and r_3 is the case where B is bound but C is free. Given the family of systems we are working with, these are the only relevant subcases and are mutually exclusive.

Recall that the activity of a rule is defined as the number of ways in which it is possible to apply the rule multiplied by the rule rate. This determines the likelihood that the rule is applied next and only depends on the current state of the system. We have not yet chosen rates for the refined rules above. In the cascade examples, it was the case that simply assigning each new rule the same rate as the original rule resulted in a neutral refinement. Even in the preceding red/blue example, under the SSA convention (§2.3) where the stochastic simulator implicitly deals with automorphisms (which *is* the case of the implementation in the Kappa Factory), the same uniform assignment of rates gives a neutral refinement.

However, if we assume this is true in this case and run a simulation of the system before and after the rule refinement with an initial state of $x(0, 100)$ and plot the activities of the rules, we see that such rate choices are not appropriate, as the sum of the activities of the refined rules is clearly different to the activity of the original rule r (Fig. 8). So setting each refined rule rate to be equal to the original does *not* result in a neutral refinement in this case.

When we look at the refined rules we see that the problem most likely lies with r_2 whose left hand side contains symmetries that should increase the ways in which it can be applied to x but which are ignored by the simulation algorithm. The question we wish to answer is how do these factors contribute to the dynamics and how, in general, do we choose rates for refined rules. The stochastic semantics of the system is determined by the joint activities of all the rules in the rule set over all states of the system. In order to preserve this stochastic semantics, we need to preserve the global activity. Intuitively, this means we must ensure that the joint activities of the rules we are adding to the system are equal to the activity of the rule we are replacing. Below we derive a general formula to choose the refined rules such that the global activity is preserved. We will return to this example when we have a general solution.

This concludes our series of examples where hopefully we have shown how natural and useful refinements are. As seen in the last couple of examples, choosing rates even for a neutral refinement is not necessarily as simple as one might imagine. Moreover, the cascade examples, particularly the ‘transistor’, illustrates just how subtle (and useful) the effects of kinetic refinement can be. In order

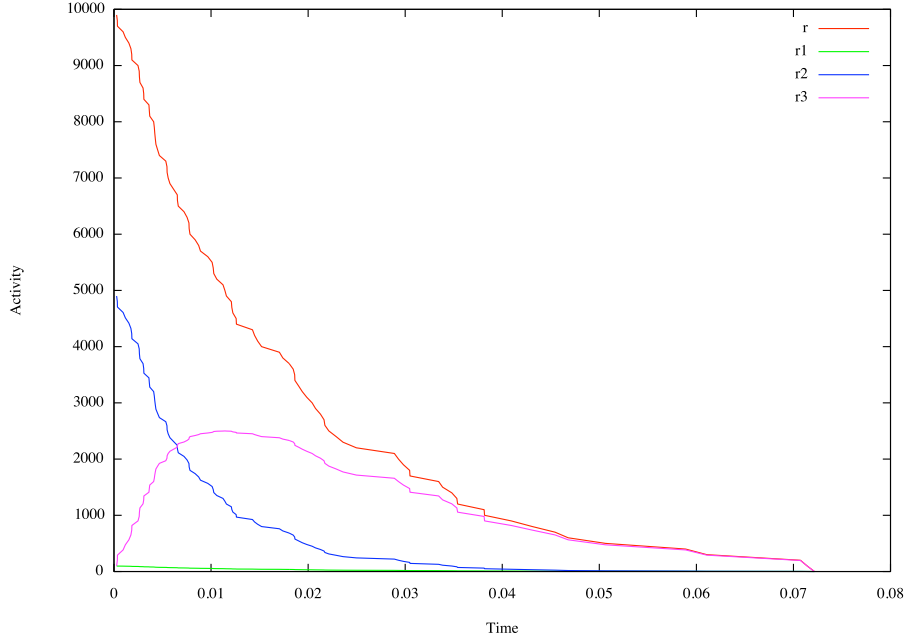


Figure 8: *Graph of all rule SSA activities with initial state $x(0, 100)$.*

to fully understand how refined rules contribute to the dynamics of the system, we have developed a formal mathematical framework to study the kappa modelling language and rule refinements. The following sections (§3, §4) present the technical aspects of this work and derive a general formula for choosing rule rates for neutral refinements that generalizes the homogeneous framework developed in [1]. This gives us a baseline reference for subsequently modulating the behaviour of a rule set with kinetic refinements, a process which is otherwise difficult to design and justify.

3 Rule-based modelling

The modelling approach we have explained informally (§1), and illustrated with examples (§2), we shall now make precise using a simple categorical language where:

- the state of a system is seen as an object x ,
- and the various ways a rule r may apply to x are seen as arrows from r 's left hand side to x .

To make this presentation easier, we simplify the Kappa syntax in two respects. The first is that we assume agents have no internal states. There is no loss of generality here, since internal states can be modelled as sites that can bind to specific unary nodes. The second simplification is that wildcard bindings (as in §2.2.1) are not used, eg expressions such as $A(x^-)$ meaning that x is bound to an unspecified agent are not considered. However, for a given set of rules it is easy to compute a syntactic super-approximation of all possible bindings (in effect defining the contact map associated to the rule set, defined below), and wildcards can be replaced with all possible concrete bindings agents they may correspond to. So, this again does not lose generality.

3.1 Notation

We will use a certain number of basic notations hereafter. If given a family of sets $(A_i; i \in I)$, we write $\sum_{i \in I} A_i$ for their disjoint sum. We write $A \setminus B$ for the set of elements of A which are not in B , $\wp(A)$ for the set of subsets of A . If an equivalence relation $\simeq$ on a set A is given, we write $A/\simeq$ for the set of equivalence classes, and $A/\simeq^* \subseteq A$ for a generic selection of representatives. If f, g are maps to a partial order, then $f \leq g$ means that f is pointwise below g . If f is a partial map on A , we write $\text{dom}(f) \subseteq A$ for its domain of definition. Finally we define a partial pairing on a set A as an irreflexive symmetric binary relation on A such that an element of A is in relation with at most one (other) element of A .

We suppose we are given a set $\mathbb{A}$ of agent names, and a set $\mathbb{S}$ of site names. A *signature* is a map from agent names $\mathbb{A}$ to sets of sites $\wp(\mathbb{S})$.

3.2 Objects and arrows

Definition 1 (objects) *An object is a quadruple $(V, \lambda, \sigma, \pi)$ where:*

- V is a finite set of nodes,
- $\lambda \in \mathbb{A}^V$ assigns agent names to nodes,
- $\sigma \in \wp(\mathbb{S})^V$ assigns sets of site names to nodes,
- π is a partial pairing over the disjoint sum $\sum_{v \in V} \sigma(v)$.

The pairing π represents bindings, or edges, between sites. Because π is a pairing, any given site can be bound at most once, but a node can be bound many times via different sites.

The simplest non-empty objects are single nodes with no sites (and therefore no binding).

We define $(u, a) \in \pi$ as shorthand for $\exists(v, b) : (u, a, v, b) \in \pi$, and say (u, a) is *free* when $(u, a) \notin \pi$, *bound* when $(u, a) \in \pi$.

Definition 2 (arrows) *An arrow from x to y is a map $f : V_x \rightarrow V_y$ such that*

- 1) f preserves names: $\lambda_y f = \lambda_x$
- 2) f preserves sites: $\sigma_y f \supseteq \sigma_x$
- 3a) f preserves edges: $(u, a, v, b) \in \pi_x \Rightarrow (f(u), a, f(v), b) \in \pi_y$
- 3b) f reflects edges: $(f(u), a) \in \pi_y, a \in \sigma_x(u) \Rightarrow (u, a) \in \pi_x$
- 4) f is injective

The category $\mathbf{S}$ of *site graphs* has objects and arrows (aka morphisms) as above. We write $[x, y]$ for the set of arrows from x to y . We say that y embeds x , or that x is embedded in y when $[x, y] \neq \emptyset$. Rules, defined below, will use arrows to recognise a partial configuration and rewrite it. Importantly the existence of an arrow in $[x, y]$ is not enough information to apply a rule, since there are many ways in which y can embed x . This is why we have to keep track of arrows. In particular, the set $[x, x]$ of automorphisms of x (aka symmetries) might contain more than the identity map, and this brings subtle aspects in rule refinement.

Our $\mathbf{S}$ lives naturally in a larger category of *contact maps* where one relaxes both the definition of objects and arrows:

- objects are as above with π any symmetric relation (no longer necessarily a partial pairing),
- and arrows are only asked to satisfy clauses 1), 2) and 3a) (no longer 3b) or 4)).

An example of a contact map is shown Fig. 9 which sums up all possible pairings that the simple cascade given in §3.1 can form. Note in particular that X 's unique site s is linked twice, so π is indeed not a pairing.

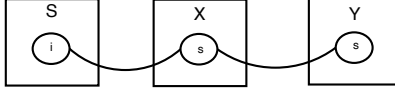


Figure 9: *The contact map of the simple cascade given in §3.1.*

One has an obvious forgetful functor from site graphs (and contact maps as well) to the category of graphs and graph morphisms -one simply forgets sites. However, from the point of view of graphs, the reflectivity condition 3b) above does not really make sense, one really needs sites to express edge reflection.

If $f \in [x, y]$, we write $f(x)$ for y the target object of f not to be confused with the image of f , $f(V_x) \subseteq V_y$. We also define the site image of f in y as $Im(f) := \{(f(v), a); v \in V_x, a \in \sigma_x(v)\}$. This is a subset of $\sum_{v \in V_x} \sigma_y(f(v))$, and only sites in $Im(f)$ are mentioned in the arrow-defining clauses above.

Conditions 2) and 3a) and the notion of pairing alone already severely constrain arrows:

Lemma 1 (rigidity) *Suppose x is connected, then any non-empty partial injection f from V_x to V_y extends to at most one morphism in $[x, y]$.*

Proof: If f is strictly partial, ie if $V_x \setminus dom(f)$ is not empty, pick $v \in V_x$ such that for some node $w \in dom(f)$, and some sites a, b , $(w, a, v, b) \in \pi_x$. This is always possible because x is connected. Then, either $(f(w), a, v', b) \in \pi_y$ for some $v' \in V_y$, and by 3a) one must extend f as $f(v) = v'$, or there is no such extension. $\square$

3.3 Extensions

Our theory of refinement (§4) relies entirely on choosing appropriate sets of extensions of a given s up to isomorphism, so it is worth examining a few basic properties thereof.

It is easy to see that an arrow is injective iff it is a mono, but there are more epis than surjections:

Lemma 2 (epis) *A map $h \in [x, y]$ is an epi iff every connected component of y intersects $f(x)$; that is to say for all connected component $c_y \subseteq y$, $h^{-1}(c_y) \neq \emptyset$.*

Proof: Suppose $f_1 h = f_2 h$ for $h \in [x, y]$, $f_i \in [y, z]$, and let $c_y \subseteq y$ be a connected component of y such that $h^{-1}(c_y) \neq \emptyset$. Pick u such that $h(u) \in c_y$, then $f_1(h(u)) = f_2(h(u))$ and by Lemma 1 $f_1/c_y = f_2/c_y$. $\square$

We write $[x, y]^e \subseteq [x, y]$ for the epis from x to y .

Definition 3 (extensions) *The category $\mathbf{S}(s)$ of extensions of an object s is defined as:*

- objects are epis $\phi \in [s, x]^e$ for some x
- an arrow $\psi : \phi_1 \rightarrow \phi_2$ is an arrow in $\mathbf{S}$ such that $\psi\phi_1 = \phi_2$

If $\psi\phi_1 = \phi_2$ then ψ is an epi, because ϕ_2 is one; ψ is also the unique such arrow, because ϕ_1 is an epi. The category of extensions of s is therefore a simple graph (meaning there is at most one arrow between any two objects); however it is *not* a partial order, two extensions ϕ_1, ϕ_2 may be isomorphic.

We write $\phi_1 \simeq_{\mathbf{S}(s)} \phi_2$ when ϕ_1 and ϕ_2 are isomorphic. It is easy to see that this is the case iff:

- there are ψ, ψ' such that $\psi\phi_1 = \phi_2$, and $\psi'\phi_2 = \phi_1$,
- or there is ψ such that $\psi\phi_1 = \phi_2$ and $\sigma_{\phi_2(s)}\psi = \sigma_{\phi_1(s)}$.

One has to be careful that saying that $\phi_1 \simeq_{\mathbf{S}(s)} \phi_2$ is a stronger statement than saying that their targets $\phi_1(s), \phi_2(s)$ are isomorphic in $\mathbf{S}$ (as site graphs). Consider the simple example where $s = A(a), A(a)$, and $t = A(a), A(a, b^1), B(a^1)$; one has two maps ϕ_1, ϕ_2 in $[s, t]$, and their targets are isomorphic, since they are equal; however they are not isomorphic as objects of $\mathbf{S}(s)$. We will return to this example later.

3.4 Actions and rules

To complete our presentation of Kappa we need to explain rules. A rule is seen as an object, an action thereon, and a rate at which the rule applies.

An atomic action on s is one of the following:

- an edge addition $+u, a, v, b$
- an edge deletion $-u, a, v, b$
- an agent addition $+A, \sigma$ with A a name, σ a set of free sites
- an agent deletion $-u$

with $u, v \in V_s$, $a \in \sigma_s(u)$, and $b \in \sigma_s(v)$.

An atomic action on s is defined:

- if $\alpha = +u, a, v, b$, when both (u, a) and (v, b) are free in s ,
- if $\alpha = -u, a, v, b$, when $(u, a, v, b) \in \pi_s$.

An action on s is a finite sequence of atomic actions on s . The notion of definedness extends recursively to non atomic actions. We consider only defined actions hereafter.

Definition 4 (rules) *A rule is a triple $r = s, \alpha, \tau$ where s is an object, α is an action defined on s , and τ is a rate (a positive real number).*

We write $\alpha \cdot s$ for the effect of the action α on s (usually called the right hand side).

Given $f \in [s, x]$ and α one defines $f(\alpha)$ the transport of an atomic α along f as:

- $f(\pm u, a, v, b) = \pm f(u), a, f(v), b$,
- $f(+A, \sigma) = +A, \sigma$,
- and $f(-u) = -f(u)$.

This can be extended to non atomic actions.

Conditions 3a) and 3b) entail that $f(\alpha)$ is defined on x when α defined on s .

Given an $f \in [s, s']$ and a rule $r = s, \alpha, \tau$, one can define the image rule $f(r) = s', f(\alpha), \tau$.

3.5 Events and probabilities

A rule set R defines a quantitative labelled transition system on objects:

Definition 5 (transitions) *Let $r = s, \alpha, \tau$ be a rule, R be a set of rules, x be an object.*

Define:

- the set of events in x associated to r as $\mathcal{E}(x, r) := \{r\} \times [s, x]$
- the set of events in x associated to R as $\mathcal{E}(x) := \sum_{r \in R} \mathcal{E}(x, r)$
- the activity of r at x is $\mathbf{a}(x, r) := \tau|[s, x]|$
- and the activity of R at x is $\mathbf{a}(x) := \sum_{r \in R} \mathbf{a}(x, r)$

This defines a labelled continuous Markov chain on objects:

$$x \xrightarrow{r}_f f(\alpha) \cdot x \tag{1}$$

where $f \in [s, x]$ and the event r, f has probability $\tau/\mathbf{a}(x)$ if $\mathbf{a}(x) > 0$, and the subsequent time advance is an exponential random variable $\delta t(x)$ such that $p(\delta t(x) > t) := e^{-\mathbf{a}(x)t}$.

As long as the quantitative structure of the transition system is determined by the activities of its rules, our analysis of refinements holds; the exact form of the dependency does not matter.

We say r, r' are isomorphic rules, written $r \simeq r'$, if there is an isomorphism $\theta \in [s, s']$ such that $r' = \theta(r)$. If that is the case then r and $\theta(r)$ have isomorphic transitions:

$$x \xrightarrow{r}_f f(\alpha) \cdot x \Leftrightarrow x \xrightarrow{\theta(r)}_{f\theta^{-1}} f\theta^{-1}(\theta(\alpha)) \cdot x$$

and in particular the same events and activities $\mathbf{a}(r, x) = \mathbf{a}(\theta(r), x)$. These rules are therefore indistinguishable.

To define activities one can use the alternate SSA convention and define $\mathbf{a}(x, r) := \tau|[s, x]|/|[s, s]|$. As discussed earlier (§2.3), this convention is natural in the special case of Petri nets because siteless objects have simple automorphisms, and rule application uses simpler contexts. In particular two injections in $[s, x]$ that are conjugated by an automorphism of s will define transitions to isomorphic states. This is no longer true with the richer contexts offered by site graphs, as the context can easily be made to distinguish conjugate injections by breaking their symmetry. The convention we choose to follow, ie to consider each injection (of each rule) as a distinct event, is more natural in our context.

4 Refinement, theoretically

Now that we have our basics in place we turn first to the question of what constitutes a refinement of a rule r . Intuitively, it is a set of extensions (in the sense of §3.3) that will partition the various ways in which a rule can be applied. To capture this notion we introduce now the concept of growth policy.

4.1 Growth policies

The idea of a growth policy for an object s is to state explicitly what sites need to be present along each extension of s . We start with a mathematical definition. Later we will investigate practical ways to obtain growth policies.

Let s be a fixed object in $\mathbf{S}$, and G be a family of maps $G(\phi)$ from $V_{\phi(s)}$ to $\wp(\mathbb{S})$ indexed by s 's extensions $\mathbf{S}(s)$. Define $\mathbf{S}(s, G)$ as the subcategory of $\mathbf{S}(s)$ obtained by restricting oneself to objects ϕ such that $\sigma_{\phi(s)} \leq G(\phi)$. Objects not in $\mathbf{S}(s, G)$ correspond to overgrown extensions where one has more sites than asked for.

Definition 6 (growth policy) *One says G is a growth policy for s if for all $\psi\phi \in \mathbf{S}(s, G)$, and for all $u \in V_{\phi(s)}$:*

$$G(\phi)(u) = G(\psi\phi)(\psi(u))$$

The above condition is called the *faithful* condition and ensures that any sites that are asked for over an extension are also asked for over further extensions.

Given a growth policy G , the set of extensions we wish to use as refinements are those that have grown fully under G , ie for all nodes in $\phi(s)$ all and only the sites asked for by G are present.

Definition 7 (refinement) Given a rule $r = s, \alpha, \tau$, and a growth policy G for s , one defines the refinement of s and r via G :

$$\begin{aligned} G(s) &:= \{\phi \in \mathbf{S}(s) \mid \sigma_{\phi(s)} = G(\phi)\} / \simeq_{\mathbf{S}(s)} \\ G(r) &:= \{\phi(r) \mid \phi \in G(s)^*\} \end{aligned}$$

where $G(s)^*$ stands for a particular selection of representatives in $G(s)$.

There are a few comments worth making about this definition.

First, the resulting $G(r)$ does not depend on the selection. Evidently if $\phi_1 \simeq_{\mathbf{S}(s)} \phi_2$, then the associated refined rules $\phi_1(r)$ and $\phi_2(r)$ are equivalent. As discussed earlier (§2.3) the rates are unchanged, as all the clever accounting is hidden in the quotient under the equivalence $\simeq_{\mathbf{S}(s)}$.

This begs the second remark, namely that $G(r)$ can contain isomorphic rules, since one is only selecting up to $\simeq_{\mathbf{S}(s)}$, and as we have noticed earlier this is a more stringent condition than having isomorphic targets. To illustrate this consider again the example where $s = A(a), A(a)$, $t = A(a), A(a, b^1), B(a^1)$, and one has two non isomorphic extensions ϕ_1, ϕ_2 in $G_{\Sigma}(s)$. Suppose the rule action is $A(a), A(a) \rightarrow A(a^1), A(a^1)$, then $\phi_1(r) \simeq \phi_2(r)$ since this action is stable under the automorphism of t that swaps the two A s. This is entirely similar to the red-blue example (§2.3).

Third, there is no reason why $G(s)^*$ should be finite, or even computable. As said, ours is a purely mathematical definition.

4.2 Simple growth policies

An example is the empty growth policy $G_{\emptyset}$ defined as:

$$G_{\emptyset}(\phi)(u) = \begin{cases} \sigma_s(v) & \text{if } u = \phi(v) \\ \emptyset & \text{if } u \notin \phi(V_s) \end{cases}$$

The faithful condition on growth policies is satisfied as only sites that exist in s are asked for over any (and all) extensions. Furthermore $G_{\emptyset}(s) = [s, s] / \simeq_{\mathbf{S}(s)}$ contains only the identity map (up to selection), and $G_{\emptyset}(r) = r$ for any r of the form s, α, τ . This is as expected, one asks for no new sites in $G_{\emptyset}$ and therefore the original rule is its own refinement.

A simple way to obtain further growth policies is to let the growth requirement of a node only depend on the node's name. This leads to what we call *homogeneous* refinements.

Specifically, given a signature Σ , one can define:

$$G_{\Sigma}(\phi)(u) := \Sigma(\lambda_{\phi(s)}(u))$$

Obviously G_{Σ} satisfies the faithful condition, since by condition 1), $\lambda_{\psi\phi(s)}(\psi(u)) = \lambda_{\phi(s)}(u)$.

The associated refinement is given by:

$$G_{\Sigma}(s) := \{\phi \mid \sigma_{\phi(s)} = \Sigma\} / \simeq_{\mathbf{S}(s)}$$

This set is not empty as soon as $\sigma_s \leq \Sigma \circ \lambda_s$. It can well be infinite. Suppose for example that $s = A()$, $\Sigma(A) = \{a, b\}$, $\Sigma(B) = \emptyset$ for $A \neq B$, then $G_{\Sigma}(s)$ consists of all pointed chains and cycles one can form with each node being of type A and having a, b as sites. Even after selection the resulting $G_{\Sigma}(s)^*$ is clearly infinite.

4.3 Neutral refinements

We can obtain now our first general result which expresses the fact that the refined rules $G(r)$ decompose unambiguously r .

4.3.1 Injectivity

Theorem 1 (injectivity) *Let $r = s, \alpha, \tau$ be a rule, G be a growth policy for s , and x an object: the composition map from $\mathcal{E}(x, G(r))$ to $\mathcal{E}(x, r)$ is injective.*

Proof: We want to prove that the map from $\sum_{\phi \in G(s)^*} [\phi(s), x]$ to $[s, x]$ mapping (ϕ, γ) to $\gamma\phi$ is injective.

So, let us suppose given two factorisations $\gamma_1\phi_1 = \gamma_2\phi_2$ as in the diagram below:

$$\begin{array}{ccc}
 u \in s & \xrightarrow{\phi_1} & \phi_1(u) = u_1 \in t_1 \\
 \phi_2 \downarrow & \nearrow \alpha & \downarrow \gamma_1 \\
 \phi_2(u) = u_2 \in t_2 & \xrightarrow{\gamma_2} & \gamma_1(u_1) = \gamma_2(u_2) = v \in x
 \end{array}$$

We want to show that $\phi_1 = \phi_2$, and $\gamma_1 = \gamma_2$. It is enough to prove that $\phi_1 = \phi_2$ because ϕ_1 is an epi, so it is enough to prove $\phi_1 \simeq_{S(s)} \phi_2$ since one selects one representative only per class in $G(s)$. But then it is enough to prove that there is an epi α such that $\alpha\phi_1 = \phi_2$, since by symmetry there will also be an α' such that $\phi_1 = \alpha'\phi_2$ and this implies the above ($\phi_1 = \alpha'\alpha\phi_1 \Rightarrow \alpha'\alpha = Id$). So consider the partial injective map from V_{t_1} to V_{t_2} defined by $\alpha := \gamma_2^{-1}\gamma_1$, we are going to prove that this map is total and an epi such that $\alpha\phi_1 = \phi_2$.

Pick $v_1 \in V_{t_1}$; we wish to prove that $\gamma_2^{-1}\gamma_1(v_1)$ is defined. Since ϕ_1 is an epi there is $u \in V_s$ such that there is a path p that connects $u_1 := \phi_1(u)$ to v_1 , and γ_1 being an arrow, there is a image path in x that connects $v := \gamma_1(u_1) = \gamma_2(u_2)$ to $\gamma_1(v_1)$. Pick along this image path the node that immediately precedes the first node (if any) that is not in $Im(\gamma_2)$ (ie where γ_2^{-1} is undefined). Call that node e (as exit), this is the node via which one exits $Im(\gamma_2)$, call a the site of e used to exit $Im(\gamma_2)$, write $e_i := \gamma_i^{-1}(e)$, and call p' the path p up to e_1 . Because γ_2 is an arrow, it must be that e_2 does not have a as a site.

Consider now the object t obtained from s by grafting p' and the associated extension $\phi \in [s, t]$. Since p' is connected to s , ϕ is indeed an epi. Also, by construction, there are (epis) β_1 and β_2 such that $\beta_i\phi = \phi_i$.

We are in a position to apply the condition on G , since all our objects ϕ, ϕ_1, ϕ_2 are in $\mathbf{S}(s, G)$, which combined with the fact that ϕ_1, ϕ_2 are in $G(s)$ gives:

$$a \in \sigma_{t_1}(e_1) = G(\phi_1)(e_1) = G(\phi)(\beta_1^{-1}(e_1)) = G(\phi_2)(\beta_2\beta_1^{-1}(e) = e_2) = \sigma_{t_2}(e_2)$$

which means e_2 has a as a site, after all. Therefore the image path of p along γ_1 wholly lies in $Im(\gamma_2)$, γ_2^{-1} is defined at $\gamma_1(v_1)$, and α is total. It is easy to verify that α is an arrow. Trivially $\alpha\phi_1 = \phi_2$, and α is hence an epi (since it is the suffix of ϕ_2). $\square$

4.3.2 Surjectivity

It is time to tackle the question of refinement surjectivity.

Given $f \in [s, x]$, and G a growth policy for s , f can be decomposed as $f = \gamma\phi$ where ϕ is maximal in $\mathbf{S}(s)$ such that $\sigma_{\phi(s)} \leq G(\phi)$. Such a ϕ is called the G -factorisation of f in x . We say x is G -decomposable if for all f its G -factorisation is in $G(s)$.

We can illustrate this definition with a growth policy we have already considered: $s = A()$, $\Sigma(A) = \{a, b\}$, $\Sigma(B) = \emptyset$ for $A \neq B$, and $G = G_\Sigma$.

Here are some examples:

- $x = A(a)$, then $\phi = f \notin G_\Sigma(s)$, and x is not G_Σ -decomposable;
- $x = A(a, b^1), B(a^1)$, then $\text{Im}(\phi) = \{A(a)\}$, so again $\phi \notin G_\Sigma(s)$ and x is not G_Σ -decomposable;
- $x = A(a, b^1), A(a^1, b), A(a, b)$, then it is easy to see that x is G_Σ -decomposable.

In the first case, x does not have enough sites, while in the second it has too many. In the third x is G_Σ -decomposable because it is itself obeying signature Σ .

Theorem 2 *Let $r = s, \alpha, \tau$ be a rule, G be a growth policy for s , and x be G -decomposable, the composition map from $\mathcal{E}(x, G(r))$ to $\mathcal{E}(x, r)$ is bijective.*

Proof: We already know the composition map is injective, so all there remains to prove is that it is surjective. But since x is G -decomposable any $f \in [s, x]$ factorises as $\gamma\phi$ with $\phi \in G(s)$. $\square$

Given R a rule set, r a rule in R , we write $R[r \setminus G(r)]$ for the rule set obtained by replacing r with $G(r)$.

We have just seen that $\mathcal{E}(x, G(r))$ and $\mathcal{E}(x, r)$ are in bijection. In other words each event $f \in [s, x]$ associated to rule $r = s, \alpha, \tau$ has a unique matching refined event γ associated to some unique rule $\phi(r) = \phi(s), \phi(\alpha), \tau$. Since $\gamma\phi = f$ both events have the same effect on x .

This establishes the following:

Corollary 1 *Let R be a rule set, and G be a growth policy, R and $R[r \setminus G(r)]$ determine the same stochastic transition system on G -decomposable states.*

4.4 Example concluded

We can now conclude our pending example where we had $s := C(), B()$, and:

$$\begin{aligned} t_1 &:= C(x^1), B(x^1) \\ t_2 &:= C(x^1), B(x^1), C(x^2), B(x^2) \\ t_3 &:= C(x^1), B(x^1), C(x) \end{aligned}$$

Observe that the above refinement are definable by a name-based growth policy where: $\Sigma(B) = \Sigma(C) = \{x\}$, and Σ is empty on all over names. The set $G_\Sigma(s)$ contains many more extensions (eg $C(x^1), C(x^1), B(x)$), but these have no matches on the states $x(n_1, n_2)$ considered here.

Now, although we have $|[s, t_2]^e| = 2$ (recall that epis must have images in all connected components), these two extensions are actually isomorphic, so the refinement of r via t_2 only contributes one rule to $G(r)$ according to Def. 7. This means that by giving r_1, r_2 and r_3 uniformly the same rate, that of the original rule, k_r , their activities will add up exactly to that of r .

However, our simulation engine follows the SSA convention (where a rule's activity is modulated by dividing by the number of automorphisms of its left hand side). So in the only case of refinement r_2 which has two automorphisms, we need to "rescale" k_{r_2} and assign it rate 2.0 to get from the SSA activity to ours. To numerically verify this, we ran a simulation of the refined system with $x(0, 100)$ as the initial state. Figure 10 shows the simulation run and we can see that at all times the refined activities add up to the original one (the top curve).

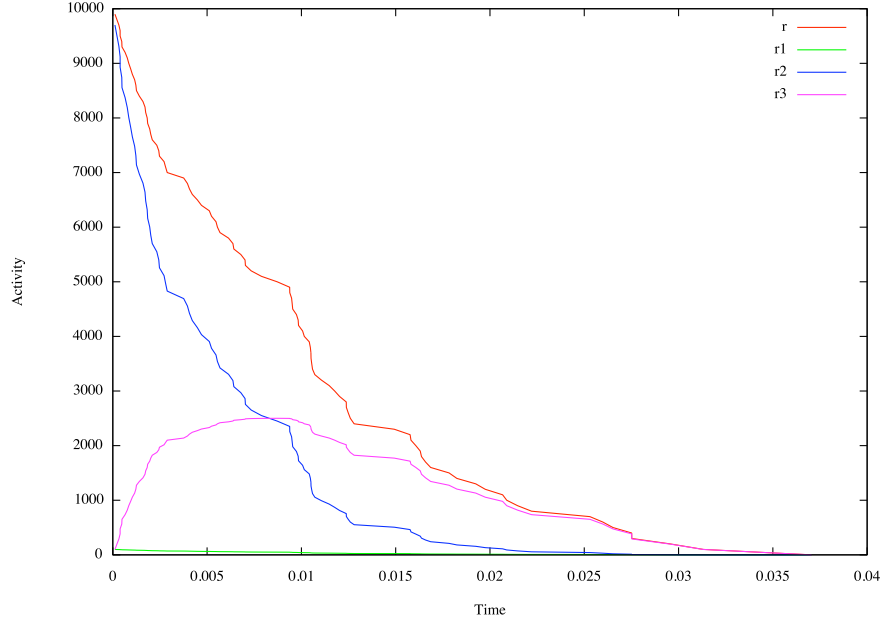


Figure 10: The activities of the refined rules r_1 , r_2 , and r_3 add up exactly to r 's (top curve).

4.5 Growth policies, concretely

We turn now to the question of how one can define concrete non-homogeneous growth policies. To do this we introduce a notion of *address*, which is a way to designate a node in an extension of s by a path that connects them to the image of a node in s .

Let us define $\mathbb{S}^{2*}$ as the set of *formal paths*, that is the set of finite even-length sequences with values in $\mathbb{S}$.

Definition 8 (addresses) *Given $\phi \in \mathbf{S}(s)$, $u \in V_s$, $u \in V_{\phi(s)}$, p a formal path, one says that (u, p) is an address for $v \in V_{\phi(s)}$ in ϕ , written $v = \llbracket u, p \rrbracket_\phi$, if p defines in $\phi(s)$ a path starting from $\phi(u)$ and ending at v .*

One says that v is addressable if it has an address; one says u, p is valid in ϕ if it addresses some v .

Clearly, for a given ϕ , an address uniquely identifies $v \in V_{\phi(s)}$, and because ϕ is an epi, every v can be addressed in this way. However, a v can be addressed in different ways. For instance, if $s = A(a^1), B(b^1)$, then $A = \llbracket A, \emptyset \rrbracket_{Id} = \llbracket B, b, a \rrbracket_{Id}$, where we have written $\emptyset$ for the empty path.

Now we want to define our growth policy for a given node in an extension based on its address, and we need only specify this for nodes that we actually want to see in a refinement. To do this we introduce the notion of formal growth and allow it to be only a partial assignment of sets of sites to addresses.

Definition 9 (formal growth) A partial map g from $V_s \times \mathbb{S}^{2*}$ to $\wp(\mathbb{S})$ is said to be a formal growth on s if for all $u \in V_s$, $(u, \emptyset) \in \text{dom}(g)$, and for all $\phi \in \mathbf{S}(s)$:

$$\llbracket u, p \rrbracket_\phi = \llbracket u', p' \rrbracket_\phi, (u, p), (u', p') \in \text{dom}(g) \Rightarrow g(u, p) = g(u', p')$$

Given a formal growth one defines:

$$G_g(\phi)(v) := \begin{cases} g(u, p) & \text{if } v = \llbracket u, p \rrbracket_\phi \text{ and } (u, p) \in \text{dom}(g) \\ \emptyset & \text{else} \end{cases}$$

The condition of Def. 9 ensures that $G_g(\phi)(v)$ does not depend on the choice of an address in the domain of g .

Lemma 3 If g is a formal growth, then G_g is a growth policy.

Proof: To see this, pick $\psi(v)$ in the codomain of an extension $\psi\phi \in \mathbf{S}(s, G_g)$.

Suppose $G_g(\phi)(v) = g(u, p)$ as in the first case of the definition above. Then, by definition, $\llbracket u, p \rrbracket_\phi = v$, and $(u, p) \in \text{dom}(g)$; certainly $\llbracket u, p \rrbracket_{\psi\phi} = \psi(v)$, so $G_g(\psi\phi)(\psi(v)) = g(u, p)$.

Suppose now one is in the second case, and v has no address in $\text{dom}(g)$. Again by definition $G_g(\phi)(v) = \emptyset$ and since $\phi \in \mathbf{S}(s, G_g)$ this forces $\sigma_{\phi(s)}(v) = \emptyset$, which ϕ being an epi is only possible if $v = \phi(u)$ for $u \in V_s$, but then an address for v is simply $(u, \emptyset) \in \text{dom}(g)$. $\square$

A similar argument shows readily that all nodes in a $\phi \in \mathbf{S}(s, G_g)$ have an address in $\text{dom}(g)$, and $G_g(s)$ is the set of extensions of s where all nodes are addressable within $\text{dom}(g)$ and have all the required sites.

We will call such growth policies address-based, or also sometimes weakly homogeneous.

It is easily seen that a name-based growth policy G_Σ is a particular type of address based growth policy. We may safely assume to simplify notations that sites can only belong to one node type (else the notion of formal path can incorporate the name of the nodes visited as well as the sites). An address u, p along any ϕ will therefore always lead to a node of the same type, namely the type of the last site of p , or that of u if $p = \emptyset$, and therefore G_Σ assigns to nodes with the same address the same growth.

Likewise it is easy to verify that the empty growth policy $G_\emptyset$ is address-based: just define $g(u, \emptyset) = \sigma_s(u)$ for $u \in V_s$. But, of course, the point of this new notion is that it is more flexible, and we shall now see an example of a weakly homogeneous refinement which is not homogeneous.

4.6 A Weakly Homogeneous Refinement

Let us consider a system with three agent types: a *Receptor* R with two sites a and m , a *Messenger* M with one site y , and an *Adaptor* A with two sites r and m . The system contact map summarising all the bindings one is interested in is shown Fig. 11.

We have a rule, say for the activation of M , with left hand side $s := My^-$, ie the rule requires that M is bound via the y site, but does not care what it is bound to. We wish to refine the rule into the following extensions:

$$\begin{aligned} t_1 &= M(y^1), A(m^1, r) \\ t_2 &= M(y^1), A(m^1, r^2), R(a^2) \\ t_3 &= M(y^1), R(x^1) \end{aligned}$$

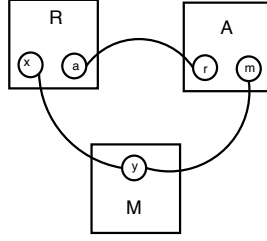


Figure 11: *Contact map for RAM system*

In each case there is clearly a unique arrow in $[s, t_i]$, so it is enough to specify the target of the extension.

A concrete motivation for this refinement would be the need to express the fact that the activation of M is impossible in the case t_1 , and faster when it is directly connected to the receptor as in t_3 . However, t_2 and t_3 both mention a node of type R with different binding sites, so the desired refinement cannot be obtained by an homogeneous refinement, meaning there is no Σ such that the above t_i s are jointly in $G_\Sigma(s)$.

So let us define a finite formal growth g as follows:

$$\begin{aligned} g(M, \emptyset) &= \{y\} \\ g(M, ym) &= \{m, r\} \\ g(M, ymra) &= \{a\} \\ g(M, yx) &= \{x\} \end{aligned}$$

where other pairs (M, p) are assumed not in $\text{dom}(g)$.

We have to verify that g is a formal growth (as in Def. 9). There are two conditions. The first holds trivially. The second also does because no two addresses in the domain of g can address the same node. Indeed, suppose $(M, p), (M, q)$ belong to the domain of g , and address the same node in some extension ϕ ; just by the type of their end nodes one sees the only non trivial possibility is $p = yx, q = ymra$ where the end node is of type R ; but $M, yx, M, ymra$ cannot both be valid in ϕ since y can only be paired with one of x and m .

The diagram below shows $\mathbf{S}(s, G_g)$ and $G_g(s)$ (Fig. 12), and one can observe that all extensions shown there project to the contact map (Fig. 11).

One can observe that the formal growth we have constructed can address any node in at most one way. In other words if $\llbracket u, p \rrbracket_\phi = \llbracket u', p' \rrbracket_\phi$, and both $(u, p), (u', p')$ are in $\text{dom}(g)$, then in fact $u, p = u', p'$. This is an easy way to satisfy the constraint of Def. 9, and it might be that this suffices in practice, at least when s is connected.

It is also worth noticing that the formal growth is a finite object. This is general. For any extension $\phi \in G(s)$, $|V_{\phi(s)}| \leq |\text{dom}(g)|$, since each node is addressable at least once within $\text{dom}(g)$. Therefore if $\text{dom}(g)$ is finite, refinements are uniformly bounded in size, and $G(s)$ is finite. That is in sharp contrast with name-based growth policies which are in general defining infinite refinements (as the earlier example of chains and rings).

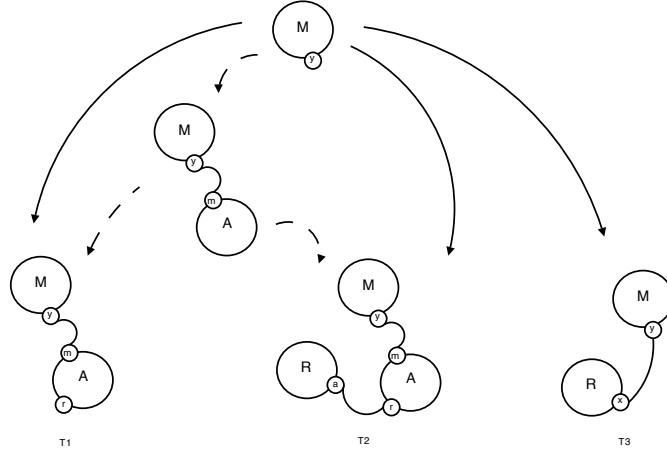


Figure 12: The refinement $G_g(s)$ of s via G_g for the RAM system (solid arrows); intermediate extensions in $\mathbf{S}(s, G_g)$ are shown too (dotted arrows).

4.7 Non-Homogeneous Growth Policies

There are ways for objects to grow that do not obey the constraints of the above examples. To see this, let us consider another example.

We are given $s := B()$ and the following contact map.

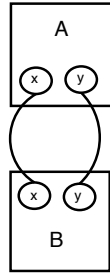


Figure 13: Contact map for the $A(x, y), B(x, y)$ system.

Clearly the following t_i define each a unique extension of s , which we will call ϕ_i :

$$\begin{aligned} t_1 &= B(x, y) \\ t_2 &= A(x^1), B(x^1, y) \\ t_3 &= B(x, y^1), A(y^1) \\ t_4 &= A(x^1), B(x^1, y^2), A(y^2) \\ t_5 &= B(x^1, y^2), A(x^1, y^2) \end{aligned}$$

There is a valid growth policy to describe this, namely:

$$\begin{aligned}
G(\phi_1)(\llbracket B, \emptyset \rrbracket_{\phi_1}) &= G(\phi_2)(\llbracket B, \emptyset \rrbracket_{\phi_2}) = \dots = G(\phi_5)(\llbracket B, \emptyset \rrbracket_{\phi_5}) = \{x, y\} \\
G(\phi_2)(\llbracket B, xx \rrbracket_{\phi_2}) &= \{x\} \\
G(\phi_3)(\llbracket B, yy \rrbracket_{\phi_3}) &= \{y\} \\
G(\phi_4)(\llbracket B, xx \rrbracket_{\phi_4}) &= \{x\}, G(\phi_4)(\llbracket B, yy \rrbracket_{\phi_4}) = \{y\} \\
G(\phi_5)(\llbracket B, xx \rrbracket_{\phi_5}) &= \{x, y\}
\end{aligned}$$

Where all other extensions, which are not intermediates of the above, are assigned uniformly $G(\phi) = \emptyset$. (Note that we have used the address notation which is convenient even for general growth policies.)

The refinements associated to G are shown Fig. 14. It is not an address-based growth policy as its formal growth would have to verify the contradictory assignments $g(B, xx) = \{x\} = \{x, y\}$ because of extensions ϕ_2 and ϕ_5 respectively.

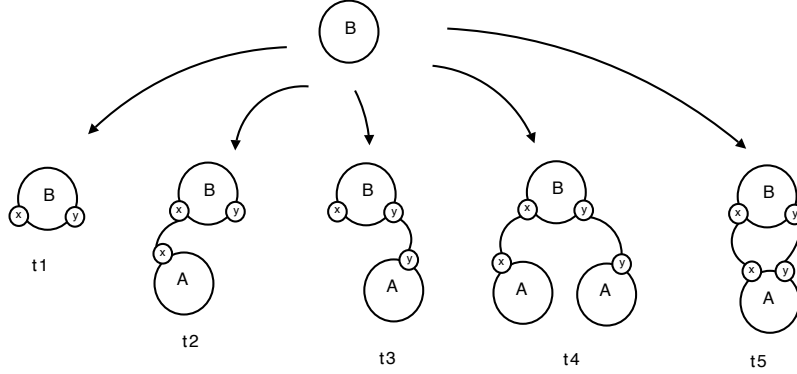


Figure 14: *Growth diagram for the cyclic AB system.*

An interesting thing to note is that if we relax the injectivity requirement in our ambient category **S** (condition 4) in Def. 2) the above is no longer a valid growth policy at all. With this requirement lifted, there is an arrow h from t_4 to t_5 mapping the two A s to the unique A in t_5 , which has sites $\{x, y\}$; this h violates the faithful condition. This shows that G being a growth policy depends on the ambient category, and may also suggest that one has a simpler refinement theory without condition 4).

5 Conclusion

We have presented a notion of refinement for rule based modelling.

Specifically we have:

- proposed a notion of growth policy to manage the growth of objects and produce refined rules
- and characterised situations where these refinements preserve the original behaviour

The somewhat technical treatment of these concepts is justified by the fact that, unlike in the case of simpler rewriting frameworks such as Petri nets, the solution to this problem needs a thorough understanding of symmetries.

Our refinements as they stand under this approach are a lot more flexible than in our previous work [1]. However, they still do not allow full flexibility. For example, it would be beneficial if we were able to use the wild card binding where we say that an agent is bound to a site not in a growth policy but we do not care what the site is. This would extend the validity of the surjectivity argument which establishes that a set of refined rules covers all cases. There is no real mathematical difficulty here, but the needed notations could prove daunting.

Another question which we have not addressed is how to generate growth policies. The growth policies used in our examples were informally generated using the contact map to keep track of possible bindings that should be added to extensions. An interesting future direction for this work would be to develop a formal theory of how to generate refinements starting with a possibly restricted notion of formal growth. This would have evident utility in a modelling environment, in that it could guarantee a user that his refinements are neutral, without him having to sit down and do the kind of calculations we have done here in a few examples.

Talking about which, we have shown on the more practical side, how refinement can be used to derive models. This of course is the main motivation for our study. This sort of approach might offer a structured top-down approach to navigate the space of parameters of a given model for data fitting. Indeed, every time one refines a rule, more parameters are made available to optimisation. This line of thought is pursued further in the recent Ref. [10] where one derives a biological repair scheme from a very simple initial model solely by means of successive refinements.

Finally, on the more conceptual side, one wonders whether refinements could provide some interesting grammatical substrate, with a measure of biological plausibility, that one could use to investigate the plasticity of existing pathways and study their evolution.

References

- [1] Vincent Danos, Jerome Feret, Walter Fontana, Russell Harmer, and Jean Krivine. Rule-based modelling, symmetries, refinements. In Springer, editor, *FMSB 2008*, volume 5054 of *LNBI*, pages 103–122, Jun 2008.
- [2] Vincent Danos, Jérôme Feret, Walter Fontana, and Jean Krivine. Abstract interpretation of cellular signalling networks. In F. Logozzo et al., editor, *VMCAI’08*, volume 4905 of *LNCS*, pages pp. 83–97. Springer, Jan 2008.
- [3] Vincent Danos. Agile Modelling of Cellular Signalling. *Computation in Modern Science and Engineering, Volume 2, Part A*, 963:611–614, Sep 2007.
- [4] Vincent Danos, Jérôme Feret, Walter Fontana, and Jean Krivine. Scalable simulation of cellular signaling networks. In Z. Shao, editor, *Proceedings of APLAS 2007*, volume 4807 of *LNCS*, pages 139–157, Nov 2007.
- [5] Vincent Danos and Linus Schumacher. How liquid is biological signalling? *TCS A*, Sep 2008.
- [6] W.S. Hlavacek, J.R. Faeder, M.L. Blinov, R.G. Posner, M. Hucka, and W. Fontana. Rules for Modeling Signal-Transduction Systems. *Science’s STKE*, 2006(344), 2006.
- [7] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *J. Phys. Chem*, 81:2340–2361, 1977.

- [8] C.Y.F. Huang and J.E. Ferrell. Ultrasensitivity in the mitogen-activated protein kinase cascade. *Proceedings of the National Academy of Sciences*, 93(19):10078–10083, 1996.
- [9] Roby P Bhattacharyya, Attila Remenyi, Brian J Yeh, and Wendell A Lim. Domains, motifs, and scaffolds: the role of modular interactions in the evolution and wiring of cell signaling circuits. *Annu Rev Biochem*, 75:655–680, 2006.
- [10] Vincent Danos, Jerome Feret, Walter Fontana, Russell Harmer, and Jean Krivine. Investigation of a biological repair scheme. In G. Paun, editor, *Proceedings of WMC'09*, volume 5391 of *LNCS*. Springer, Jan 2009.